```php
<?php

//Require base class for derivation
require_once ("Server.php");

//Define class as derivative of HTTP_WebDAV_Server
class HTTP_WebDAV_Server_MySQL extends HTTP_WebDAV_Server
{
        /**
         * Root directory for WebDAV access
         *
         * Defaults to webserver document root (set by ServeRequest)
         *
         * @access private
         * @var    string
         */
         var $base = "/";

    /**
     * MySQL Host
     *
     * @access private
     * @var    string
     */
     var $db_host = "db01";

    /**
     * MySQL database
     *
     * @access private
     * @var    string
     */
     var $db_name = "15272";

    /**
     * MySQL user for db access
     *
     * @access private
     * @var    string
     */
     var $db_user = "15272";

    /**
      * MySQL password for db access
      *
      * @access private
      * @var    string
      */
      var $db_passwd = "3366";


    /**
      * Serve a webdav request
      *
      * @access public
      * @param  string
      */
      function ServeRequest($base = false)
      {
                //Litmus compliance
                foreach (apache_request_headers() as $key => $value) {
                        if (stristr($key, "litmus")) {
                                error_log("Litmus test $value");
                                header("X-Litmus-reply: ".$value);
                        }
                }
```

```
        // establish connection to db
        mysql_connect($this->db_host, $this->db_user, $this->db_passwd)
        or die(mysql_error());
        mysql_select_db($this->db_name)
        or die(mysql_error());

        // let the base class do all the work
        parent::ServeRequest();
      }


    /**
      * Authentication of users from db
      *
      * @access private
      * @param  string  HTTP Authentication type (Basic, Digest, ...)
      * @param  string  Username
      * @param  string  Password
      * @return bool    true on successful authentication
      */
      function check_auth($type, $user, $pass)
      {
                /* TODO: test and enable this feature
                $query = "SELECT * FROM users WHERE username = '$user' AND password =
password('$pass') LIMIT 1";
                $result = mysql_query($query);

                if(  mysq_num_rows($result) == 1 )
                {*/
                        return true;
                /*}else{
                        return false;
                }*/
      }

      //Check if dir exists
      function CheckDir($Path , $SQLReturn = "ownerid")
      {
                $Path = $this->_slashify($Path);
                //Does exist
                $query = "SELECT $SQLReturn AS sqlreturn FROM dirs WHERE path = '$Path' limit 1";
                $CheckDirResult = mysql_query($query) or die(mysql_error());

                //Does exist
                if(  mysql_num_rows($CheckDirResult) != 1 )
                {
                        $ToReturn =  false;
                }else{
                        $row = mysql_fetch_array($CheckDirResult);
                        $ToReturn = $row["sqlreturn"];
                }
                mysql_free_result($CheckDirResult);
                return $ToReturn;
      }

      //Check if file exists
      function CheckFile($Path)
      {
                $Path = $this->_unslashify($Path);
                //Does exist
                $query = "SELECT ownerid FROM files WHERE path = '$Path' limit 1";
                $result = mysql_query($query);
                //Does exist
                if(  mysql_num_rows($result) != 1 )
                {
                        $ToReturn = false;
```

```
                }else{
                        $ToReturn = true;
                }
                return $ToReturn;
        }

        //Creates dir and sub dirs
        function CreateDir($Path)
          {
                if( !$this->CheckDir($Path) )
                {
                        if($Ownerid = $this->CheckDir($this->GetParentDir($Path)))
                        {
                        $result = mysql_query("INSERT INTO dirs (path,ownerid) VALUES
('$Path','$Ownerid')");
                                if( $result )
                                        $ToReturn = true;
                                else
                                        $ToReturn = false;
                                }
                        else
                        {
                                $ToReturn = $this->CreateDir($this->GetParentDir($Path));
                        }
                }
                else
                {
                        $ToReturn = true;
                }
                return $ToReturn;
          }

            //Gets the sub directory of a path
        function GetParentDir($Path)
            {
                $Path = $this->_unslashify($Path);
                $Levels = explode("/",$Path);

                $Levels[sizeof($Levels)-1] = "";

                $ParentDir = implode("/",$Levels);
                return $this->_slashify($ParentDir);
            }

        function GetMimetype($Path)
          {
          //TODO: Use magic mimetype if available on server instead of this...

                /*
                Include list of known mimetypes, the complete list is available from IANA, this is
just a small peace of it...
                */
                require("MimetypeList.php");

                //Explode the path to extract file extension
                $levels = explode(".", $this->_unslashify($Path));

                //Grab the last record of the array
                $fileext = $levels[sizeof($levels)-1];

                //Can we detect the mimetype
                if (array_key_exists($fileext,$ext))
                {
                        //if yes, detect and return mimetype
                        return $ext[$fileext];
                }else
```

```
                {
                        //If false return mimetype undetectable
                        return "application/octet-stream";

                        /*
                        I følge RFC2046 afsnit 4.1.4 side 11 skal ikke genkendte minetypes
håndteres som application/octet-stream. Se evt. IANA.
                        */
                }
        }

     /**
      * GET method handler
      *
      * @param  array  parameter passing array
      * @return bool   true on success
      */
      function GET(&$options)
      {
                //Is file
                $query = "SELECT contentid,lastmodified,mimetype,filesize FROM files WHERE path =
'{$this->_unslashify($options['path'])}' limit 1";
                $result = mysql_query($query);
                if(  mysql_num_rows($result) != 1 )
                {

                        $ownerid = 0;
                        $ownerid = $this->CheckDir($options['path'],"id");

                        //Does exist
                        if( $ownerid == 0 )
                        {
                                $ToReturn = false;
                        }else{
                                //Return directory index, in readable format!


                                //Do not check this is path = /, _unslashify will crash
                                if($options["path"] == "/")
                                {
                                        //If we aren't requesting a directory that exist
                                        $path = $this->_slashify($options["path"]);
                                        if ($path != $options["path"]) {
                                                return false;
                                        }
                                }

                                // fixed width directory column format
                                $format = "%15s  %-19s  %-s\n";

                                echo "<html><head><title>Index of
".htmlspecialchars($this->_slashify($options['path'])."</title></head>\n";

                                echo "<h1>Index of
".htmlspecialchars($this->_slashify($options['path'])."</h1>\n";

                                echo "<pre>";
                                printf($format, "Size", "Last modified", "Filename");
                                echo "<hr>";

                                $dirquery = "SELECT path,lastmodified FROM dirs WHERE ownerid =
'$ownerid'";
                                $filequery = "SELECT path,lastmodified,filesize FROM files WHERE
ownerid = '$ownerid'";

                                $dirresult = mysql_query($dirquery);
```

```
                                if(  mysql_num_rows($dirresult) >= 1 )
                                {
                                        while($list = mysql_fetch_array($dirresult))
                                        {
                                                $name =
htmlspecialchars($this->_slashify(basename($list["path"])));
                                                printf($format,
                                                        "-",
                                                        strftime("%Y-%m-%d %H:%M:%S",
strtotime($list["lastmodified"])),
                                                        "<a href='$name'>$name</a>"
                                                        );
                                        }
                                }

                                $fileresult = mysql_query($filequery);

                                if(  mysql_num_rows($fileresult) >= 1 )
                                {
                                        while($list = mysql_fetch_array($fileresult))
                                        {
                                                $name = htmlspecialchars(basename($list["path"]));
                                                printf($format,
                                                        number_format($list["filesize"]),
                                                        strftime("%Y-%m-%d %H:%M:%S",
strtotime($list["lastmodified"])),
                                                        "<a href='$name'>$name</a>"
                                                        );
                                        }
                                }
                                echo "</pre>";

                                echo "</html>\n";
                                exit;
                        }
                }else{
                        $row = mysql_fetch_array($result);

                        $query2 = "SELECT * FROM binarycontent WHERE id = '{$row["contentid"]}'
limit 1";
                        $result2 = mysql_query($query2);

                        //do NOT handle file if no file content is available
                        if( mysql_num_rows($result2) != 1)
                        {
                                $ToReturn = false;
                        }else{

                                // detect resource type
                                $options['mimetype'] = $row["mimetype"];

                                // detect modification time
                                // see rfc2518, section 13.7
                                // some clients seem to treat this as a reverse rule
                                // requiering a Last-Modified header if the getlastmodified header
was set
                                $options['mtime'] = strtotime($row["lastmodified"]);

                                // detect resource size
                                $options['size'] = $row["filesize"];

                                $row2 = mysql_fetch_array($result2);

                                // no need to check result here, it is handled by the base class
                                $options['data'] = $row2["content"];
```

```
                                $ToReturn = true;
                        }
                }
                return $ToReturn;
        }


    /**
      * MKCOL method handler
      *
      * @param  array  general parameter passing array
      * @return bool   true on success
      */
          function MKCOL($options)
          {
                $parent = $this->GetParentDir($options["path"]);
                $name = basename($path);

                /*TODO figure out why filesystem version uses this?
                if (!file_exists($parent)) {
                return "409 Conflict";
                }*/

                //Do not create an existing directory
                if(!$this->CheckDir($options["path"]))
                {
                        //If parent isn't a directory
                        if (!$this->CheckDir($parent))
                        {
                                $ToReturn = "409 Conflict";
                        }else{

                                //if files exist where you want to place dir
                                if ( $this->CheckFile($this->_unslashify($options["path"])))
                                {
                                        $ToReturn = "405 Method not allowed";
                                }else{
                                        if (!empty($_SERVER["CONTENT_LENGTH"]))
                                        {
                                                // no body parsing yet
                                                $ToReturn = "415 Unsupported media type";
                                        }else{

                                                $result = mysql_query("SELECT id FROM dirs WHERE
path = '$parent'");

                                                $ownerid = mysql_result($result, 0);

                                                $query = "INSERT INTO dirs
(ownerid,path,creationdate,lastmodified) VALUES
('$ownerid','{$this->_slashify($options["path"])}',NOW(),NOW())";
                                                $result = mysql_query($query);

                                                if (!$result)
                                                {
                                                        $ToReturn = "403 Forbidden";
                                                }else{
                                                        $ToReturn = "201 Created";
                                                }
                                        }


                                }
                        }
                }else{
                        $ToReturn = "405 Method not allowed";
```

```
                }
                return $ToReturn;
            }

           /**
         * PUT method handler
         *
         * @param  array  parameter passing array
         * @return bool   true on success
         */
        function PUT(&$options)
        {
                if($this->CheckDir($options["path"]))
                {
                        $options["new"] = "409 Conflict";
                }else{
                        if($options["new"] != "403 Forbidden")
                        {
                                if($this->CheckFile($options['path']))
                                {
                                        $query = "SELECT contentid FROM files WHERE path =
'{$options['path']}' limit 1";
                                        $Results = mysql_query($query); //or die(mysql_error());

                                        if( mysql_num_rows($Results) == 1 )
                                        {
                                                $contentid = mysql_result($Results, 0);
                                                $filesize = strlen($options["data"]);
                                                //Update
                                                $Result1 = mysql_query("UPDATE binarycontent SET
content = '{$options["data"]}' WHERE id = '$contentid'");

                                                        $Resutl2 = mysql_query("UPDATE files SET
filesize = '$filesize', lastmodified = NOW() WHERE path = '{$options["path"]}'");
                                        }else{
                                                //TO DO handle error!
                                        }
                                        $options["new"] = false;
                                }else{
                                        $this->CreateDir($this->GetParentDir($options['path']));


                                        /*$query = "SELECT id FROM dirs WHERE path =
'{$this->GetParentDir($options['path'])}' limit 1";
                                        $Results = mysql_query($query);
                                        $row = mysql_fetch_array($Results);
                                        $ownerid = $row["id"];*/
                                        $ownerid =
$this->CheckDir($this->GetParentDir($options['path']),"id");


                                        $InsertQuery = "INSERT INTO binarycontent
(content)VALUES('{$options["data"]}')";
                                        mysql_query($InsertQuery);

                                        $filesize = strlen($options["data"]);
                                        $contentid = mysql_insert_id();
                                        $query = "INSERT INTO files
VALUES('$ownerid','{$options["path"]}',NOW(),NOW(),'{$this->GetMimetype($options["path"])}','$files
ize','$contentid')";
                                        mysql_query($query);
                                        $options["new"] = true;
                                }
                        }
                }
```

```
                $ToReturn = $options["new"] ? "201 Created" : "204 No Content";
                return $ToReturn;
        }


    /**
      * DELETE method handler
      *
      * @param  array  general parameter passing array
      * @return bool   true on success
      */
      function DELETE($options)
      {
                $id = 0;
                $id = $this->CheckDir($options["path"],"id");
                if($id != 0){
                        $this->DelDir($options["path"]);
                        /*$delquery = "DELETE FROM dir WHERE ownerid = $id";
                        mysql_query($delquery);
                        $delquery = "DELETE FROM files WHERE ownerid = $id";
                        mysql_query($delquery);*/
                        $ToReturn = "204 No Content";
                }else{
                        if($this->CheckFile($options["path"]))
                        {
                                $this->DelFile($options["path"]);
                                $ToReturn = "204 No Content";
                        }else{
                                $ToReturn = "404 File not found";
                        }
                }
                return $ToReturn;
          }

    /**
      * PROPFIND method handler
      *
      * @param  array  general parameter passing array
      * @param  array  return array for file properties
      * @return bool   true on success
      */
        function PROPFIND(&$options, &$files)
        {
                $files["files"] = array();

                $query = "SELECT id,creationdate,lastmodified,path FROM dirs WHERE path =
'{$this->_slashify($options['path'])}' limit 1";
                $result = mysql_query($query);
                if(  mysql_num_rows($result) != 1 )
                {
                        $ToReturn = false;
                }else{

                        $row = mysql_fetch_array($result);
                        $info = array();

                        $info["path"] = $this->_slashify($row["path"]);
                        $info["props"] = array();
                        $info["props"][] = $this->mkprop("displayname",strtoupper($row["path"]));
                        $info["props"][] = $this->mkprop("creationdate",
strtotime($row["creationdate"]));
                        $info["props"][] = $this->mkprop("getlastmodified",
strtotime($row["lastmodified"]));
                        $info["props"][] = $this->mkprop("resourcetype", "collection");
                        $info["props"][] = $this->mkprop("getcontenttype", "httpd/unix-directory");
```

```
                        $files["files"][] = $info;

                        //If content of containing resource is requested?
                        if(!empty($options["depth"]))
                        {
                                //Setting path id
                                $PathId = $row["id"];

                                $query = "SELECT path,creationdate,lastmodified FROM dirs WHERE
ownerid = '$PathId'";
                                $result = mysql_query($query);
                                if(  mysql_num_rows($result) > 0 )
                                {
                                        while($row = mysql_fetch_array($result))
                                        {
                                                $info = array();
                                                $info["path"] = $this->_slashify($row["path"]);
                                                $info["props"] = array();
                                                $info["props"][] =
$this->mkprop("displayname",strtoupper($row["path"]));
                                                $info["props"][] = $this->mkprop("creationdate",
 strtotime($row["creationdate"]));
                                                $info["props"][] =
$this->mkprop("getlastmodified",strtotime($row["lastmodified"]));
                                                $info["props"][] = $this->mkprop("resourcetype",
"collection");
                                                $info["props"][] = $this->mkprop("getcontenttype",
"httpd/unix-directory");
                                                $files["files"][] = $info;
                                        }
                                }

                                $query = "SELECT path,creationdate,lastmodified,mimetype,filesize
FROM files WHERE ownerid = '$PathId'";
                                $result = mysql_query($query);
                                if(  mysql_num_rows($result) > 0 )
                                {
                                        while($row = mysql_fetch_array($result))
                                        {
                                                $info = array();
                                                $info["path"] = $this->_unslashify($row["path"]);
                                                $info["props"] = array();
                                                $info["props"][] =
$this->mkprop("displayname",strtoupper($row["path"]));
                                                $info["props"][] = $this->mkprop("creationdate",
 strtotime($row["creationdate"]));
                                                $info["props"][] =
$this->mkprop("getlastmodified", strtotime($row["lastmodified"]));
                                                $info["props"][] = $this->mkprop("resourcetype",
"");
                                                $info["props"][] = $this->mkprop("getcontenttype",
$row["mimetype"]);
                                                $info["props"][] =
$this->mkprop("getcontentlength", $row["filesize"]);;
                                                $files["files"][] = $info;
                                        }
                                }
                        }

                        $ToReturn = true;
                }
                return $ToReturn;
        }


        function DelDir($Path)
```

```
	{
		$id = false;
		$id = $this->CheckDir($Path,"id");
		if($id != false)
		{
			//Check if it contains subdirs
			$CQuery = "SELECT path FROM dirs WHERE ownerid = '$id'";
			$CResult = mysql_query($CQuery);

			//Delete any subdir
			if(mysql_num_rows($CResult) >= 1)
			{
				while($SubDirs = mysql_fetch_array($CResult))
				{
					$this->DelDir($SubDirs["path"]);
				}
			}
			mysql_free_result($CResult);

			//Find any stored files in dir
			$CQuery = "SELECT path FROM files WHERE ownerid = '$id'";
			$CResult = mysql_query($CQuery);

			//Delete any File in directory
			if(mysql_num_rows($CResult) >= 1)
			{
				while($Files = mysql_fetch_array($CResult))
				{
					$this->DelFile($Files["path"]);
				}
			}
			mysql_free_result($CResult);

			$delquery = "DELETE FROM dirs WHERE id = '$id' limit 1";
			mysql_query($delquery);

			$ToReturn = true;
		}
		else
		{
			$ToReturn = true;
		}
		return $ToReturn;
	}

	function DelFile($Path)
	{
		if($this->CheckFile($Path))
		{

			$Path = $this->_unslashify($Path);
			$FQuery = "SELECT contentid from files where path = '$Path' limit 1";
			$Fresult = mysql_query($FQuery);

			if(mysql_num_rows($Fresult) != 0)
			{
				$row = mysql_fetch_array($Fresult);
				$delquery = "DELETE FROM binarycontent WHERE id =
'{$row["contentid"]}' limit 1";
				mysql_query($delquery);
			}

			$delquery = "DELETE FROM files WHERE path = '$Path' limit 1";
			mysql_query($delquery);

			$ToReturn = true;
```

```
                }else{
                        $ToReturn = false;
                }
                return $ToReturn;
        }

          //Function that writes to debug.log, used for debuggin information.
        function printtodebug($msg,$description)
        {
                //Access debug.log to append data
                $debug = fopen("./debug.log","a");

                ob_start();
                ob_start();  // add the inner buffering envelope
                session_start();
                print_r($msg);
                ob_end_flush(); // closing the inner envelope will activate URL rewriting
                $msg = ob_get_contents();
                ob_end_clean();

                $msg = "\n\n ******* $description ******** \n$msg";

                //Return boolean instead of msg length
                if(fwrite($debug,$msg))
                {
                        return true;
                }else{
                        return false;
                }
        }
}

?>
```